SIAM J. COMPUT.
Vol. 28, No. 4, pp. 1247–1263

EFFICIENT GENERATION OF MINIMAL LENGTH ADDITION CHAINS*

EDWARD G. THURBER†

Abstract. An addition chain for a positive integer n is a set $1 = a_0 < a_1 < \cdots < a_r = n$ of integers such that for each $i \geq 1$, $a_i = a_j + a_k$ for some $k \leq j < i$. This paper is concerned with some of the computational aspects of generating minimal length addition chains for an integer n. Particular attention is paid to various pruning techniques that cut down the search time for such chains. Certain of these techniques are influenced by the multiplicative structure of n. Later sections of the paper present some results that have been uncovered by searching for minimal length addition chains.

Key words. addition chain, search tree, pruning bounds, backtracking, branch and bound

AMS subject classifications. 11Y16, 11Y55, 68Q25

PII. S0097539795295663

1. Introduction.

Dellac [6] asks what is the minimum number of multiplications required to raise a number A to the power m. Since exponents are added when powers of the same base are multiplied, this gives rise to the concept of an addition chain. This topic has been studied somewhat extensively over the intervening 100 years. Many questions have been posed concerning this deceptively simple problem. While some of these have been answered, others remain tantalizingly open.

This paper explores some of the computational aspects of generating minimal length addition chains for an integer n. The emphasis on computation is of interest in light of the surprising discoveries that the generation of minimal length addition chains has uncovered, some of which led Knuth [12] to say that perhaps no conjecture concerning addition chains is safe. Some of these discoveries will be mentioned in section 10.

The algorithm to be explored for generating minimal length addition chains is a backtracking algorithm that uses branch and bound methods to prune the search tree. Since the algorithm is exponential in nature, particular attention will be paid to pruning the search tree. The pruning bounds dramatically increase the efficiency of the search and increase the feasibility of pursuing a variety of questions concerning addition chains. For example, the determination of $c(r)$, the first integer requiring r steps in an addition chain of minimal length, is greatly facilitated by these methods.

Of course, the trick when pruning a search tree is not to cut off too much. Establishing the validity of the pruning bounds used in the backtracking algorithm will be a primary focus of what follows. It will be seen that certain classes of pruning bounds are affected by the multiplicative structure of n.

The paper is organized as follows. Section 2 includes some notation and prior results. Section 3 develops the notion of the search tree for addition chains. In section 4, an algorithm is discussed which will find all the minimal addition chains for the integer n. The algorithm is adapted easily to find just one such chain. In section 5, the pruning bounds that come most readily to mind are developed. In

*Received by the editors December 6, 1995; accepted for publication (in revised form) September 16, 1997; published electronically March 22, 1999.
http://www.siam.org/journals/sicomp/28-4/29566.html
†Department of Math and Computer Science, Biola University, La Mirada, CA 90639 (ed@irvine.com).

section 6, improvements are made to class 1 bounds. Section 7 discusses pruning bounds that involve more than one member of an addition chain. In section 8, class 1 and class 2 bounds are applied to n and their effectiveness is explored. In section 9, class 1 pruning bounds are refined in a way that is influenced by the multiplicative nature of n, and a summary is given of the pruning bounds developed in the paper. Section 10 further explores the efficiency of the various classes of pruning bounds and presents some computational results found concerning the addition chain problem. Finally, in section 11 concluding remarks are made with some reference to future directions.

2. Preliminaries. An addition chain for a positive integer n is a set $1 = a_0 < a_1 < \cdots < a_r = n$ of integers such that for every $i \geq 1$, $a_i = a_j + a_k$ for some $k \leq j < i$. The minimal length, r, of an addition chain for n is denoted by $l(n)$. As in Knuth [12], $\lambda(n) = \lfloor \log_2 n \rfloor$, and $\nu(n)$ will denote the number of ones in the binary representation of n. The function NMC(n) was introduced by the author [19] and denotes the number of minimal addition chains for n. For $n = 29$, a minimal addition chain is $1, 2, 4, 8, 9, 13, 16, 29$. In base 2, $29 = 11101_2$. Thus, $\nu(29) = 4, \lambda(29) = \lfloor \log_2 29 \rfloor = 4$, $l(29) = 7$, and, as it turns out, NMC(29) = 132.

As Knuth observed, either $\lambda(a_i) = \lambda(a_{i-1})$ or $\lambda(a_i) = \lambda(a_{i-1}) + 1$. In the former case, step i is called a *small step* and is called a *big step* otherwise. There are exactly $\lambda(n)$ big steps in any chain for n. The number of steps, r, in an addition chain for n can be expressed as $r = \lambda(n) + N(n)$, where $N(n)$ denotes the number of small steps in the chain. It should be noted that $N(n)$ is chain dependent. Minimizing $N(n)$ will result in a minimal length addition chain for n. If $j = i - 1$, then step i is called a *star step*. An addition chain that consists entirely of star steps is called a *star chain*. If $j = k = i - 1$, then step i is called a *doubling*.

Theoretically developed lower bounds for $l(n)$ provide starting values from which to start looking for minimal length addition chains. It is conjectured [12] that $l(n) \geq \lambda(n) + \lceil \log_2 \nu(n) \rceil$. If $\nu(n) \geq 2^m + 1$, the conjecture states that $l(n) \geq \lambda(n) + m + 1$. The conjecture has been established for $m = 0, 1, 2, 3$ [17], and it is known that the conjecture holds for all $n \leq$ 327,678. Schönhage [14] has shown that $l(n) \geq \log_2 n + \log_2 \nu(n) - 2.13$. Thus, $l(n) \geq \lceil \log_2 n + \log_2 \nu(n) - 2.13 \rceil$ in any event.

3. The search tree. A tree organization for the solution space for finding addition chains for n is as follows:

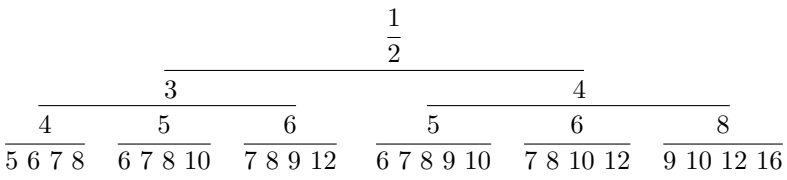

FIG. 3.1.

This tree organization is found in a paper by Chin and Tsai [5] in which they develop algorithms for finding minimal or near minimal addition chains for integers n. The tree shall be referred to as the search tree. The algorithm in this paper traverses the search tree in a fashion similar to that found in [5]. The search is depth first and considers larger numbers first (i.e., all paths that start $1, 2, 4$ are taken before any path that starts $1, 2, 3$). In what follows the emphasis is on developing and establishing the

validity of pruning bounds used in the branch and bound method to greatly speed up the search for minimal addition chains.

If $n = 1$ is at *level* 0, then 2 is at *level* 1, 3 and 4 are at *level* 2, etc. The children of a given node a_i at level i are all numbers $a_{i+1} > a_i$ formed as sums $a_j + a_k$, $k \le j \le i$ of numbers in the path from 1 to a_i. Any addition chain for an integer n can be found by taking the path from 1 down to the appropriate occurrence of n in the search tree. The tree grows exponentially as can be seen by the fact that each integer at level k in the tree has at least $k + 1$ children. This means that the number of paths to level k is at least $k!$ To find $l(n)$ and NMC(n) is to find on what level n appears first and how many times it appears on this level. If $1 = a_0, a_1, \ldots, a_i$ is a partial chain, then br$[a_i]$ is the branch of the search tree that has a_i for its root.

4. Algorithm for finding all minimal addition chains. The backtracking algorithm for finding all minimal addition chains for an integer n starts by setting $r = \lambda(n) + \lceil \log_2 \nu(n) \rceil$ if $n \le 327{,}678$ or $\nu(n) \le 16$. Otherwise, it sets $r = \lceil \log_2 n + \log_2 \nu(n) - 2.13 \rceil$. If no chain of length r is found, then set r to $r + 1$ and repeat. The process is continued until a length is tried for which chains for n exist. This length will be $l(n)$. By the binary chain method [12], $l(n) \le \lambda(n) + \nu(n) - 1$.

Starting with minimum estimates for $l(n)$ and working up is preferable to over estimating $l(n)$ and working down since it minimizes the number of small steps in the addition chains being generated. It is the increase in small steps that adds significantly to the search time.

In what follows, *lb*, which stands for lower bound, will be used in place of r. The search commences with a lower bound *lb* for $l(n)$ and increases this value incrementally if necessary. The following algorithm traverses the search tree and develops partial addition chains $1 = a_0, a_1, \ldots, a_i$ into minimal length addition chains for n. A stack is maintained which holds the possible children of a_i at each stage. The children of a_i constitute a stack segment.

Algorithm find_minimum_chains(n).
begin
  if $n \le 327{,}678$ **or** $\nu(n) \le 16$ **then**
    $lb = \lambda(n) + \lceil \log_2 \nu(n) \rceil$
  else
    $lb = \lceil \log_2 n + \log_2 \nu(n) - 2.13 \rceil$
  end if
  $a_0 = 1$, $a_1 = 2$ (first two elements of an addition chain)
  loop *lb*_value
    determine pruning bounds
    $i = 1$
    loop find_chains
      if $i < lb$ **then**
        determine whether to retain a_i
        if a_i is retained **then**
          stack the possibilities for a_{i+1} in increasing order in next stack
            segment (all sums $a_j + a_k > a_i, k \le j < i + 1$)
          increment i by 1
          let a_i be the element on top of the stack
          if $a_i = n$ **then**
            chain is found

```
            back up in the search tree until a node is found that needs further
              expanding (i.e., take the next element off the stack that is not
              in the stack segment of a_i)
          end if
        else
          back up in the search tree until a node is found that needs further
            expanding (i.e., take the next element off of the stack)
          let a_i be the element on top of the stack
        end if
      else
        back up in the search tree until a node is found that needs further
          expanding (i.e., take the next element off of the stack that is not
          in the stack segment of a_i)
        let a_i be the element on top of the stack
      end if
    end loop find_chains
    if no chains found then
      increase lb by 1
    else
      exit loop lb_value
    end if
end loop lb_value.
```

When an element a_i is taken off of the top of the stack, it is either n, in which case a minimal addition chain for n has been found, or it becomes part of a partial chain $1 = a_0, a_1, \ldots, a_i$ provided that it is not pruned from the search tree. The possible children a_{i+1} of a_i are put on the stack in increasing order. Some pruning can take place at this point, also. Backing up in the search tree is accomplished by taking elements off the stack. The stack is popped until an element is found that cannot be pruned. If, for instance, it is determined that a_i is to be pruned and that this exhausts all possibilities from a_{i-1}, then the next element on the stack is the next child of a_{i-2} in decreasing order (if children of a_{i-2} still remain; otherwise, backtrack further). The next child of a_{i-2} (the new a_{i-1}) is added to the partial chain $1 = a_0, a_1, \ldots, a_{i-2}$ provided that it is not pruned, and its children are added to the stack. When the algorithm backs up to $a_1 = 2$, it will terminate since the $1, 2, 4$ and $1, 2, 3$ branches (br[4] and br[3]) will have been traversed, and these are the only possibilities from level 2 in the search tree. At this point, either all the minimal addition chains for n will have been found or none will have been found, in which case lb is increased by 1 and the process is repeated. The algorithm will exit loop find_chains when the stack is empty. A slight variation of the algorithm will terminate once the first addition chain for n is found.

It should be mentioned that this algorithm is particularly well suited for finding all the minimal length addition chains for n. If one were interested in finding only one such chain, other strategies might be employed such as limiting the search to star chains at first and checking only for nonstar chains if no star chains are found. The pruning bounds, however, apply to whatever strategy is employed for traversing the search tree.

Two classes of bounds will be discussed. The first class (class 1 bounds) concerns bounds for a_i, while the second class (class 2 bounds) concerns bounds for $a_i + a_{i-1}$. Class 1 bounds will be refined to further improve the efficiency of the search. Finally,

the somewhat curious phenomenon will be explored of how class 1 bounds can be improved for integers not divisible by integers of the form $2^i + 1$.

5. Pruning bounds (class 1). Since $a_i \le 2a_{i-1}$ for $1 \le i \le r$, it follows that if a_j and a_i are two members of an addition chain such that $a_i > 2^m a_j$, then it will require more than m steps to get from a_j to a_i no matter how the chain is constructed. This can be used as follows to prune the search tree.

The subscript, i, of an integer a_i in an addition chain for n represents the number of steps that have been taken to reach a_i. If a chain of a certain length, $lb = i + m$, is being sought for an integer n, and if $2^m a_i < n$, then it will be impossible to get to n from a_i in $m = lb - i$ steps. The branch br$[a_i]$ of the search tree emanating from a_i can be eliminated. If a chain of length 4 is being sought for $n = 16$, then the partial chain $1, 2, 3$ cannot lead to a 4-step chain for 16 since $2^2 3 < 16$. This eliminates br[3] which cuts out 12 possible paths to level 4 (or roughly half of the 25 paths to this level).

If lb has been set, then there will be $lb - i$ steps in the chain from a_i to n if such a chain exists. If $2^{lb-i} a_i < n$, then no chain for n which includes a_i exists of length lb, and the branch br$[a_i]$ can be pruned from the search tree. It follows that if $a_i < n/2^{lb-i}$, then br$[a_i]$ is pruned from the tree. This eliminates at least $lb!/i!$ paths to search.

If $a_i < \lceil n/2^{lb-i} \rceil$, there are two cases for the integer a_i. If $n/2^{lb-i}$ is an integer, then $a_i < n/2^{lb-i}$, and if it is not an integer, then $a_i < \lceil n/2^{lb-i} \rceil$ implies that $a_i \le \lfloor n/2^{lb-i} \rfloor$. Since $n/2^{lb-i}$ is not an integer, it follows that $a_i < n/2^{lb-i}$. In any event, if $a_i < \lceil n/2^{lb-i} \rceil$, then br$[a_i]$ can be pruned from the tree. This leads to the set of class 1 bounds

$$b_i = \lceil n/2^{lb-i} \rceil, \qquad i = 0, \ldots, lb. \tag{A}$$

We have the following theorem.

THEOREM 1. *Let $\{b_i\}$ denote bounding sequence* (A) *for a positive integer n. If for some step i in an addition chain for n, $a_i < b_i$, then the partial chain $a_0, a_1, \ldots, a_i$ cannot lead to a chain of length lb for n, and br$[a_i]$ can be pruned from the search tree.*

For example, suppose $n = 39 = 100111_2$. Then $lb = \lambda(39) + \lceil \log_2 \nu(39) \rceil = 5 + 2 = 7$. The set of bounds $\{\lceil 39/2^{7-i} \rceil\} = \{1, 1, 2, 3, 5, 10, 20, 39\}$. The partial chain $1, 2, 3, 5, 8, 9$ cannot lead to a chain of length 7 for 39 since $a_5 = 9 < 10 = b_5$.

At each stage in an addition chain's development, the possible choices for the next element a_{i+1} after a_i are added as a new stack segment. This includes all a_{i+1} such that $a_{i+1} = a_j + a_k$ for $0 \le k \le j \le i$ and $a_i < a_{i+1} \le n$. In the partial chain $1, 2, 3, 5, 8, 13$ for 39 the possible choices for the next element in the chain are $14, 15, 16, 18, 21$, and 26. The bound associated with the sixth step in the chain is 20. Since $18 < 20$, it can be discarded, as can $14, 15$, and 16. The numbers 21 and 26 are added to the stack.

The bounds $\{\lceil n/2^{lb-i} \rceil\}$ can be found by dividing n by 2 and each successive result by 2. At each stage the result is rounded up to the nearest integer if necessary. This follows from the fact that $\lceil \lceil n/2^{lb-i} \rceil / 2 \rceil = \lceil n/2^{lb-i+1} \rceil$. If $a_i \ge b_i = \lceil n/2^{lb-i} \rceil$, br$[a_i]$ is retained and all possibilities for a_{i+1} such that $a_i < a_{i+1} \le n$ and $a_{i+1} \ge b_{i+1} = \lceil n/2^{lb-(i+1)} \rceil$ are stored for future consideration.

6. Class 1 bounds refined. If n is not a power of 2, the bounds $\{\lceil n/2^{lb-i} \rceil\}$ can be improved by the following considerations. For an odd integer n, the last step

in an addition chain for n is $n = a_r = a_{r-1} + a_k$ for some $k < r-1$. The last step must be a star step in a minimal chain since otherwise, a_{r-1} could be eliminated, resulting in a shorter chain. Also, $k < r-1$, or else n would be even. Thus, if $r = lb$, then $n \le a_{lb-1} + a_{lb-2}$, which implies that $n \le 3a_{lb-2}$.

It follows that a_{lb-2} cannot lead to a chain of lb steps for n unless $a_{lb-2} \ge \lceil n/3 \rceil$. a_{lb-2} will not be greater than or equal to $\lceil n/3 \rceil$ unless $a_{lb-3} \ge \lceil n/(3 \cdot 2) \rceil$, $a_{lb-4} \ge \lceil n/(3 \cdot 2^2) \rceil$, etc. Thus, br$[a_i]$ is retained only when $a_i \ge \lceil n/(3 \cdot 2^{lb-(i+2)} \rceil$ for $i = 0, \ldots, lb-2$, and $a_{lb-1} \ge \lceil n/2 \rceil$. The bounds are

$$\text{(B)} \qquad b_i = \begin{cases} \lceil n/(3 \cdot 2^{lb-(i+2)}) \rceil & 0 \le i \le lb-2, \\ \lceil n/2^{lb-i} \rceil & lb-1 \le i \le lb. \end{cases}$$

For $n = 39$, the bounds are $\{1, 1, 2, 4, 7, 13, 20, 39\}$ which is a significant improvement over bounding sequence (A).

Regardless of whether n is even or odd, it can be expressed uniquely as $n = 2^t m$, where m is odd and $t \ge 0$. Bounding sequence (B) generalizes to

$$\text{(C)} \qquad b_i = \begin{cases} \lceil n/(3 \cdot 2^{lb-(i+2)} \rceil & 0 \le i \le lb-t-2, \\ \lceil n/2^{lb-i} \rceil & lb-t-1 \le i \le lb. \end{cases}$$

This leads to the following theorem.

THEOREM 2.[1] *Let $\{b_i\}$ denote bounding sequence* (C) *for a positive integer n. If for some step i in an addition chain for n, $a_i < b_i$, then the partial chain $a_0, a_1, \ldots, a_i$ cannot lead to a chain of length lb for n, and br$[a_i]$ can be pruned from the search tree.*

Proof. Suppose $n = 2^t m$ where m is odd. Bounding sequence (C) can be split into two parts.

Region 1. $n, n/2, n/2^2, \ldots, n/2^{t-1}, m = n/2^t, \lceil m/2 \rceil$.

These are the bounds in reverse order corresponding to steps i such that $lb-t-1 \le i \le lb$.

Region 2. $\lceil m/3 \rceil, \lceil m/(3 \cdot 2) \rceil, \ldots, \lceil m/(3 \cdot 2^{lb-t-(i+2)}) \rceil = \lceil n/(3 \cdot 2^{lb-(i+2)}) \rceil, \ldots$.

These are the bounds in reverse order corresponding to steps i such that $0 \le i \le lb-t-2$.

In Region 1, $b_i = \lceil n/2^{lb-i} \rceil$, while in Region 2, $b_i = \lceil n/(3 \cdot 2^{lb-(i+2)}) \rceil$. This latter bound is determined by noting that $b_{lb-t-2} = \lceil m/3 \rceil, b_{lb-t-3} = \lceil m/(3 \cdot 2) \rceil$, $\ldots, b_{lb-t-j} = \lceil m/(3 \cdot 2^{j-2}) \rceil = \lceil n/(3 \cdot 2^{t+j-2}) \rceil, \ldots$. If $i = lb-t-j$, then $t+j-2 = lb-(i+2)$.

For Region 1, if $a_i < \lceil n/2^{lb-i} \rceil$, then br$[a_i]$ can be pruned from the search tree by the same reasoning used in Theorem 1.

For Region 2, suppose $a_i < \lceil m/(3 \cdot 2^{lb-t-(i+2)}) \rceil = \lceil n/(3 \cdot 2^{lb-(i+2)}) \rceil$. Then $a_i < n/(3 \cdot 2^{lb-i-2})$. Assume n is not a power of two since, if it is a power of two, Region 2 does not apply. Then every step in a chain for n cannot be a doubling. Suppose step s is a nondoubling in which case $a_s \le a_{s-1} + a_{s-2}$. If $s > i+1$, then

$$\begin{aligned} a_s &\le a_{s-1} + a_{s-2} \le 2^{(s-1)-i} a_i + 2^{(s-2)-i} a_i \\ &= 2^{(s-2)-i}(2a_i + a_i) = 2^{(s-2)-i}(3a_i). \end{aligned}$$

[1] Note that in this theorem and following theorems, a_i will often be replaced by $2a_{i-1}$ in a set of inequalities since $a_i \le 2a_{i-1}$. More generally, for $j < i, a_i$ will be replaced by $2^{i-j}a_j$ since $a_i \le 2^{i-j}a_j$. Additional inequalities such as $4a_i + a_{i-1} \le 3(a_i + a_{i-1})$ also follow from the fact that $a_i \le 2a_{i-1}$. These and other similar inequalities will be used frequently.

It follows that $n = a_{lb} \le 2^{lb-s}a_s \le 2^{lb-s}2^{(s-2)-i}(3a_i) < (2^{lb-2-i})(3)(n/(3\cdot 2^{lb-i-2}) = n$. Of course $n < n$ is a contradiction. Thus, there can be no nondoubling step in the chain after step $i+1$. If every step in the chain after step $i+1$ is a doubling, then $n = a_{lb} = 2^{lb-(i+1)}a_{i+1}$. This means that 2^{lb-i-1} divides n. Since $i \le lb - t - 2$, it follows that $lb - i - 1 \ge t + 1$. This is a contradiction since t is the highest power of 2 dividing n.

Thus, for any a_i in Region 2 which is less than the corresponding bound, br$[a_i]$ can be pruned from the search tree. □

7. Pruning bounds (class 2). More can be done by way of pruning the search tree by considering the sum $a_i + a_{i-1}$. It is often the case that if this sum is less than b_{i+1} of bounding sequence (C), then br$[a_i]$ can be pruned from the search tree. Bounding sequences for $a_i + a_{i-1}$ will be called class 2 bounds. (Note: The same bounding sequence, when used for a_i, is called a class 1 bounding sequence and, when used for $a_i + a_{i-1}$, is called a class 2 bounding sequence.) It is a somewhat curious fact that if n is a multiple of 5, then bounding sequence (C) must be replaced by bounding sequence (A) when used as a class 2 bound. First, the case for odd n will be considered. Bounding sequence (B) is used when n is odd, and n is not a multiple of 5. It is convenient to split it into three regions.

Region 1. $n, \lceil n/2 \rceil$.

Region 2. $\lceil n/3 \rceil$.

Region 3. $\lceil n/(3\cdot 2) \rceil, \ldots, \lceil n/(3\cdot 2^{lb-(i+2)}) \rceil, \ldots$.

In Region 1, b_{i+1} is either n or $\lceil n/2 \rceil$; that is, $i = lb - 1$ or $i = lb - 2$.

$\boldsymbol{i = lb-1}$. In this case, $a_i + a_{i-1} < b_{i+1}$ implies that $a_{lb-1} + a_{lb-2} < b_{lb} = a_{lb} = n$. Thus, n must be $a_{lb-1} + a_{lb-1} = 2a_{lb-1}$. This contradicts the fact that n is odd.

$\boldsymbol{i = lb - 2}$. If $a_{lb-2} + a_{lb-3} < b_{lb-1} = \lceil n/2 \rceil$, then $a_{lb-2} + a_{lb-3} < n/2$. Since n is odd, $n \le a_{lb-1} + a_{lb-2} \le 2a_{lb-2} + 2a_{lb-3} < 2(n/2) = n$ which is a contradiction.

For Region 2, $i = lb - 3$. If $a_{lb-3} + a_{lb-4} < b_{lb-2} = \lceil n/3 \rceil$, then $a_{lb-3} + a_{lb-4} < n/3$. We assume that $a_i \ge b_i$ for $i = 1, \ldots, n$, since it has been shown previously (Theorem 2) that if $a_i < b_i$ for any i, then a chain of lb steps for n is not possible. This means that $a_{lb-2} \ge n/3$. Since $a_{lb-3} + a_{lb-4} < n/3$, it follows that $a_{lb-2} = 2a_{lb-3}$. Step $lb - 1$ must be a star step since step $lb - 2$ is a doubling. The possibilities for a_{lb-1} need to be considered. These can be drawn from the set $\{a_{lb-2} + a_j, j \le lb - 2\}$. In what follows, $n \le a_{lb-1} + a_{lb-2}$ since n is odd, and $a_{lb-2} = 2a_{lb-3}$.

(i) $a_{lb-1} = a_{lb-2} + a_{lb-4}$:

$$n = a_{lb} \le a_{lb-1} + a_{lb-2} = 4a_{lb-3} + a_{lb-4} \le 3(a_{lb-3} + a_{lb-4}) < 3(n/3) = n.$$

Clearly, no possibilities $a_{lb-2} + a_j, j < lb - 4$ need be considered.

(ii) $a_{lb-1} = a_{lb-2} + a_{lb-3}$:

Since n is odd and step lb is a star step, the possibilities for step lb can be drawn from the set $\{a_{lb-1} + a_j, j \le lb - 2\}$.

(iia) If $n = a_{lb-1} + a_{lb-2}$, then $n = 5a_{lb-3}$. Thus, 5 divides n. If $n = 95 = 1011111_2$, then $\lambda(95) = 6$, and $\nu(95) = 6$. $lb = \lambda(95) + \lceil \log_2 \nu(95) \rceil = 9$. The bounding sequence $\{b_i\}$ is $\{1, 1, 1, 2, 4, 8, 16, 32, 48, 95\}$. An addition chain for 95 is $1, 2, 3, 5, 8, 11, 19, 38, 76, 95$. Note that $a_{lb-3} + a_{lb-4} = 19 + 11 = 30 < 32 = b_{lb-2}$. Thus, when n is divisible by 5, there often exist minimal addition chains for n when $a_{lb-3} + a_{lb-4} < b_{lb-2}$.

(iib) If $n = a_{lb-1} + a_{lb-3}$, this leads to the contradiction that $n < n$ by similar reasoning as that used in (i). Again no possibilities $n = a_{lb-1} + a_j, j < lb - 3$ need be considered.

(iii) $a_{lb-1} = a_{lb-2} + a_{lb-2}$:
If $n = a_{lb-1} + a_{lb-2}$, then $n = 3(2a_{lb-3})$ is even, and if $n = a_{lb-1} + a_j, j < lb-3$, then $n < n$ as before. Since lb must be a star step, the only possibility is $n = a_{lb-1} + a_{lb-3}$ in which case $n = 5a_{lb-3}$. n is divisible by 5 and, as in case (iia), minimal chains for n often exist when $a_{lb-3} + a_{lb-4} < b_{lb-2}$.

For Region 3, $i \le lb-4$. If $a_i + a_{i-1} < b_{i+1} = \lceil n/(2^{lb-i-3} \cdot 3) \rceil$, then $a_i + a_{i-1} < n/(2^{lb-i-3} \cdot 3)$. It follows that

$$\begin{aligned} a_{lb-3} + a_{lb-4} &< 2^{(lb-3)-i} a_i + 2^{(lb-4)-(i-1)} a_{i-1} \\ &= 2^{(lb-3)-i}(a_i + a_{i-1}) < n/3. \end{aligned}$$

It follows from the same arguments as used for Region 2 that $\mathrm{br}[a_i]$ will be pruned from the search tree unless n is a multiple of 5.

These considerations establish the following theorem.

THEOREM 3. *Let n be odd, n not a multiple of 5, and let $\{b_i\}$ be bounding sequence* (B). *If $a_i + a_{i-1} < b_{i+1}$ for some i, then the partial chain $1 = a_0, a_1, \ldots, a_i$ cannot lead to a minimal addition chain for n.*

Comment. From Theorem 1, it is known that $a_i \ge b_i$ if $1 = a_0, a_1, \ldots, a_i$ is to lead to a minimal chain for n. The following example shows that additional branches can be pruned from the search tree when $a_i + a_{i-1} < b_{i+1}$.

For $n = 39, \{b_i\} = \{1, 1, 2, 4, 7, 13, 20, 39\}$. A partial chain $1, 2, 3, 5, 7$ satisfies the requirement that $a_i \ge b_i$. However, $a_4 + a_3 = 7 + 5 < 13 = b_5$. Thus, $\mathrm{br}[7] = \mathrm{br}[a_4]$ can be pruned from the search tree.

A generalization. Theorem 3 generalizes to the case where $n = 2^t m$ and m is odd. In this case, bounding sequence (C) is used, and it is convenient to split it into the following regions.

Region 1. $n, n/2, n/2^2, \ldots, n/2^{t-1}, m = n/2^t, \lceil m/2 \rceil$.

Region 2. $\lceil m/3 \rceil$.

Region 3. $\lceil m/(3 \cdot 2) \rceil, \ldots, \lceil m/(3 \cdot 2^{lb-t-(i+2)} \rceil = \lceil n/(3 \cdot 2^{lb-(i+2)}) \rceil, \ldots$.

THEOREM 4. *Suppose n is not a multiple of 5 and $\{b_i\}$ is bounding sequence* (C). *If $a_i + a_{i-1} < b_{i+1}$ for some i and $n \ne 2^{lb-i} a_i$ for i in Region 1, then the partial chain $1 = a_0, a_1, \ldots, a_i$ cannot lead to a minimal addition chain for n.*

Note. The condition that $n \ne 2^{lb-i} a_i$ for i in Region 1 is necessary as can be seen from the following example. $1, 2, 3, 5, 7, 14, 21, 42, 84, 168, 336$ is a minimal chain for $n = 336$. Bounding sequence (C) for 336 is $1, 1, 2, 4, 7, 11, 21, 42, 84, 168, 336$. Even though $a_6 + a_5 = 21 + 14 < 42 = b_7, \mathrm{br}[a_6]$ cannot be pruned from the search tree.

The proof of Theorem 4, while tedious, is analogous to the proof of Theorem 3, with $lb - t - i$ replacing $lb - i, i \ge 0$. Also, some step $s \ge lb - t$ will need to be a nondoubling or else 2^{t+1} will divide n, which contradicts the fact that 2^t is the highest power of 2 dividing n. The element a_s in Theorem 4 plays a role similar to that of n in Theorem 3 since they are both formed by nondoublings.

Proof.

Region 1. $lb - t - 2 \le i \le lb - 1$. Note that if $i \ge lb - t - 2$, then $i + 1 \ge lb - t - 1$, and b_{i+1} will be in Region 1. If $a_i + a_{i-1} < b_{i+1} = \lceil n/2^{lb-(i+1)} \rceil$, then suppose there exists a step $s > i$ that is not a doubling. Then

$$a_s \le a_{s-1} + a_{s-2} \le 2^{(s-1)-i} a_i + 2^{(s-2)-(i-1)} a_{i-1} = 2^{(s-1)-i}(a_i + a_{i-1}).$$

It follows that

$$n = a_{lb} \le 2^{lb-s} a_s \le 2^{lb-s} 2^{(s-1)-i}(a_i + a_{i-1}) < 2^{(lb-1)-i}(n/2^{(lb-1)-i}) = n.$$

Thus, every step after step i must be a doubling. This means that $n = 2^{lb-i}a_i$ if the partial chain $1 = a_0, a_1, \ldots, a_i$ can be extended to a chain of length lb for n.

Region 2. $i = lb - t - 3$ for $b_{i+1} = \lceil m/3 \rceil$.

If $a_{lb-t-3} + a_{lb-t-4} < \lceil m/3 \rceil = \lceil n/(2^t \cdot 3) \rceil$, then $a_{lb-t-3} + a_{lb-t-4} < n/(2^t 3)$. Since it is assumed that $a_i \geq b_i$ for $i = 1, \ldots, n$, then $a_{lb-t-2} \geq b_{lb-t-2} = \lceil n/(2^t \cdot 3) \rceil \geq n/(2^t 3)$, and it follows that $a_{lb-t-2} = 2a_{lb-t-3}$. Since step $lb-t-2$ is a doubling, step $lb-t-1$ must be a star step; that is, $a_{lb-t-1} = a_{lb-t-2} + a_k$ for some $k \leq lb-t-2$. It will be shown that $k = lb-t-2$ or $k = lb-t-3$. Suppose that $k \leq lb-t-4$ and that there is a nondoubling after step $lb-t-1$. Let s be the first such step. Then $a_{lb-t-1} \leq a_{lb-t-2} + a_{lb-t-4}$ and by using inequalities as noted in footnote 1 of Theorem 2, it follows that

$$\begin{aligned} a_s &\leq a_{s-1} + a_{s-2} \leq 2^{(s-1)-(lb-t-1)}a_{lb-t-1} + 2^{(s-2)-(lb-t-2)}a_{lb-t-2} \\ &= 2^{s-lb+t}(a_{lb-t-1} + a_{lb-t-2}) \leq 2^{s-lb+t}(2a_{lb-t-2} + a_{lb-t-4}) \\ &< 2^{s-lb+t}3(n/(2^t 3)) = 2^{s-lb}n. \end{aligned}$$

Thus, $n = a_{lb} \leq 2^{lb-s}a_s < 2^{lb-s}2^{s-lb}n = n$. This means that all steps after step $lb-t-1$ are doublings which, as noted, is not possible. Thus, either $a_{lb-t-1} = a_{lb-t-2} + a_{lb-t-3}$ or $a_{lb-t-1} = a_{lb-t-2} + a_{lb-t-2}$.

So far, if $a_{lb-t-3} + a_{lb-t-4} < n/(2^t 3)$, then it has been established that:

(1) $a_{lb-t-2} = 2a_{lb-t-3}$; and

(2) $a_{lb-t-1} = a_{lb-t-2} + a_{lb-t-3}$ or $a_{lb-t-1} = a_{lb-t-2} + a_{lb-t-2}$.

Suppose $\boldsymbol{a_{lb-t-1} = a_{lb-t-2} + a_{lb-t-3}}$. Then $a_{lb-t-1} = 3a_{lb-t-3}$. There must be a step $s \geq lb-t$ that is a nondoubling or as shown before 2^{t+1} divides n. Let s be the first nondoubling after step $lb-t-1$. The possibilities for step s that have a chance can be drawn from the set $\{a_{s-1} + a_{s-2}, a_{s-1} + a_{s-3}, 2a_{s-2}\}$.

(i) If $a_s = a_{s-1} + a_{s-3}$, then $a_s \leq 2^{(s-1)-(lb-t-1)}a_{lb-t-1} + 2^{(s-3)-(lb-t-3)}a_{lb-t-3}$. It follows by inequalities, as noted in footnote 1 that

$$\begin{aligned} a_s &\leq 2^{s-lb+t}(a_{lb-t-1} + a_{lb-t-3}) = 2^{s-lb+t}(4a_{lb-t-3}) \\ &< 2^{s-lb+t}3(a_{lb-t-3} + a_{lb-t-4}) < 2^{s-lb+t}3(n/(2^t 3)) = 2^{s-lb}n. \end{aligned}$$

Thus, $n = a_{lb} \leq 2^{lb-s}a_s < 2^{lb-s}2^{s-lb}n = n$.

(ii) If $a_s = a_{s-1} - a_{s-2}$, then if $\boldsymbol{s = lb - t}$, it follows that $a_{lb-t} = a_{lb-t-1} + a_{lb-t-2}$. Since $a_{lb-t-1} = a_{lb-t-2} + a_{lb-t-3}$ and $a_{lb-t-2} = 2a_{lb-t-3}$, this means that $a_{lb-t} = 5a_{lb-t-3}$. If there is a step $h > lb-t$ which is a nondoubling, then by reasoning similar to that discussed before, it can be shown that $n < n$. This means that all the steps after step $lb-t$ must be doublings. Thus

$$n = a_{lb} = 2^{lb-(lb-t)}a_{lb-t} = 2^t(5a_{lb-t-3}).$$

This contradicts the fact that n is not a multiple of 5.

Now, suppose $\boldsymbol{s > lb - t}$. This means that $a_{lb-t} = 2a_{lb-t-1}$, since step s is the first nondoubling after step $lb-t-1$. The reasoning used before concerning nondoubling steps can be combined with the fact that $a_{lb-t-3} + a_{lb-t-4} < n/(2^t 3)$ to show that $a_s < 2^{(s-lb+t)}3(n/(2^t 3)) = 2^{(s-lb)}n$, and $n = a_{lb} \leq 2^{lb-s}a_s < n$.

(iii) If $a_s = 2a_{s-2}$, then $a_s \leq 2(2^{(s-2)-(lb-t-2)}a_{lb-t-2}) = 2^{s-lb+t+1}a_{lb-t-2} = 2^{s-lb+t}(4a_{lb-t-3})$. As in case (i), this results in $n < n$.

Suppose $\boldsymbol{a_{lb-t-1} = 2a_{lb-t-2}}$. Since $a_{lb-t-2} = 2a_{lb-t-3}$, it follows that $a_{lb-t-1} = 4a_{lb-t-3}$. There must be a first step $s \geq lb-t$ that is a nondoubling or else 2^{t+1}

divides n as before. The possibilities for step s are $\{a_{s-1}+a_{s-2}, a_{s-1}+a_{s-3}, a_{s-1}+a_{s-4}, 2a_{s-2}\}$. It will become evident that these are the only cases that need to be considered.

(i) $a_s = a_{s-1}+a_{s-4} \le 2^{(s-1)-(lb-t-1)}a_{lb-t-1} + 2^{(s-4)-(lb-t-4)}a_{lb-t-4}$
$= 2^{s-lb+t}(4a_{lb-t-3}+a_{lb-t-4}) \le 2^{s-lb+t}3(a_{lb-t-3}+a_{lb-t-4})$
$< 2^{s-lb+t}3(n/(2^t3))$.

Thus, $a_s < 2^{s-lb}n$, and $n = a_{lb} \le 2^{lb-s}a_s < 2^{lb-s}2^{s-lb}n = n$.

(ii) $a_s = 2a_{s-2}$ leads to $n < n$ as in (iii) of the case where $a_{lb-t-1} = a_{lb-t-2} + a_{lb-t-3}$.

(iii) $a_s = a_{s-1}+a_{s-2}$. This is the most difficult case, since this is the largest possible value of a_s for the nondoubling step which must occur somewhere between steps $lb-t$ and lb. In what follows, $a_{lb-t-1} = 2a_{lb-t-2}$ and $a_{lb-t-2} = 2a_{lb-t-3}$. Suppose $\boldsymbol{s = lb-t}$. Then $a_{lb-t} = a_{lb-t-1}+a_{lb-t-2} = 6a_{lb-t-3}$. If there are no more nondoublings in the chain after step $lb-t$, then $n = a_{lb} = 2^{lb-(lb-t)}a_{lb-t} = 2^t6a_{lb-t-3}$ which means 2^{t+1} divides n. Thus, there must be a first nondoubling step $h > s$. The possibilities for a_h are drawn from the set $\{a_{h-1}+a_{h-2}, a_{h-1}+a_{h-3}, 2a_{h-2}\}$.

(iiia) $a_h = a_{h-1}+a_{h-3} \le 2^{(h-1)-(lb-t)}a_{lb-t} + 2^{(h-3)-(lb-t-2)}a_{lb-t-2}$
$= 2^{h-1-lb+t}6a_{lb-t-3} + 2^{h-1-lb+t}2a_{lb-t-3} = 2^{h-lb+t}4a_{lb-t-3}$.

From this it follows, as in prior cases, that $n < n$.

(iiib) $a_h = 2a_{h-2} \le 2(2^{(h-2)-(lb-t-1)}a_{lb-t-1} = 2^{h-lb+t}4a_{lb-t-3}$ which leads to $n < n$.

(iiic) $a_h = a_{h-1}+a_{h-2}$:

$$a_h = a_{h-1}+a_{h-2} \le 2^{(h-1)-(lb-t)}a_{lb-t} + 2^{(h-2)-(lb-t-1)}a_{lb-t-1}$$
$$= 2^{h-1-lb+t}6a_{lb-t-3} + 2^{h-1-lb+t}4a_{lb-t-3} = 2^{h-lb+t}5a_{lb-t-3}.$$

The 5 in the bound is not sufficient to conclude that $n < n$. If there is another step $l > h$ which is a nondoubling, then, as before, it can be shown that $a_l \le 2^{l-lb+t}4a_{lb-t-3}$ from which it follows that $n < n$. It follows that $n = a_{lb} = 2^{lb-h}a_h = 2^{lb-h}(a_{h-1}+a_{h-2})$. If $h > lb-t+1$, then

$$n = 2^{lb-h}(a_{h-1}+a_{h-2}) \le 2^{lb-h}(2^{(h-1)-(lb-t)}a_{lb-t} + 2^{(h-2)-(lb-t)}a_{lb-t})$$
$$= 2^{lb-h}(2^{h-1-lb+t}6a_{lb-t-3} + 2^{h-2-lb+t}6a_{lb-t-3})$$
$$= 2^{t-1}9a_{lb-t-3} \le 2^{t-1}6(a_{lb-t-3}+a_{lb-t-4}) < 2^t3(n/(2^t3)) = n.$$

If $h = lb-t+1$, then $n = 2^{lb-(lb-t+1)}(a_{lb-t}+a_{lb-t-1}) = 2^{t-1}10a_{lb-t-3}$.

From this it follows that 5 divides n.

Now we continue with the case $a_{lb-t-1} = 2a_{lb-t-2}$, where step s is the first nondoubling after step $lb-t-1$ and $a_s = a_{s-1}+a_{s-2}$. Suppose $\boldsymbol{s > lb-t}$; then step $s-1$ is a doubling. If there are no more nondoublings after step s, then

$$n = 2^{lb-s}a_s = 2^{lb-2}(a_{s-1}+a_{s-2}) = s^{lb-s}(3a_{s-2})$$
$$= 2^{lb-s}3(2^{(s-2)-(lb-t-1)}a_{lb-t-1}) = 2^{t-1}3a_{lb-t-1} = 2^{t-1}3(4a_{lb-t-3}).$$

This implies that 2^{t+1} divides n, which is a contradiction. As in the previous case, let step h be the first nondoubling step after step s. The only possibility for a_h that does not lead to $n < n$ is $a_h = a_{h-1}+a_{h-2}$. As in case (iiic), it can be shown that there can be no more nondoublings after step h or else $n < n$. Thus, $n = 2^{lb-h}a_h = 2^{lb-h}(a_{h-1}+a_{h-2})$. The cases $h = s+1$ and $h > s+1$ are handled separately.

Since $s > lb - t$, step $lb - t$ is a doubling, and $a_{lb-t} = 8a_{lb-t-3}$.

If $h = s + 1$, it can be shown by reasoning similar to that used before that $n = 2^t 5 a_{lb-t-3}$, which means that 5 divides n while if $h > s + 1$; then

$$\begin{aligned} n &= 2^{lb-h}a_h = 2^{lb-h}(a_{h-1} + a_{h-2}) = 2^{lb-h}(2^{h-1-s}a_s + 2^{h-2-s}a_s) \\ &= 2^{lb-h}2^{h-2-s}(3a_s) = 2^{lb-2-s}3(a_{s-1} + a_{s-2}) \\ &= 2^{lb-2-s}3(2^{(s-1)-(lb-t)}a_{lb-t} + 2^{(s-2)-(lb-t-1)}a_{lb-t-1}) \\ &= 3(2^{t-3}12a_{lb-t-3}) \le 3(2^{t-3}8(a_{lb-t-3} + a_{lb-t-4})) < 2^t 3(n/(2^t 3)) = n. \end{aligned}$$

(iv) $a_s = a_{s-1} + a_{s-3}$. As before, it can be shown that if there is a step $h > s$ that is a nondoubling, then $n < n$. Thus, $n = 2^{lb-s}a_s = 2^{lb-s}(a_{s-1} + a_{s-3})$. The cases $s = lb - t$, $s = lb - t + 1$, $s = lb - t + 2$, and $s \ge lb - t + 3$ must be considered. In all cases it can be shown that 5 divides n. It is important to note that s is the first step after $lb - t - 1$ that is a nondoubling. For instance, if $s = lb - t + 2$, then

$$\begin{aligned} n &= 2^{lb-s}(a_{s-1} + a_{s-3}) = 2^{lb-(lb-t+2)}(a_{lb-t+1} + a_{lb-t-1}) \\ &= 2^{t-2}(16a_{lb-t-3} + 4a_{lb-t-3}) = 2^t(5a_{lb-t-3}). \end{aligned}$$

Region 3. $i < lb - t - 3$.

For this region, the bound $a_i + a_{i-1} < b_{i+1}$ translates into $a_i + a_{i-1} < \lceil m/(3 \cdot 2^{lb-t-(i+3)} \rceil = \lceil n/(3 \cdot 2^{lb-(i+3)}) \rceil$ which implies $a_i + a_{i-1} < n/(2^{lb-i-3}3)$. It follows that

$$\begin{aligned} a_{lb-t-3} + a_{lb-t-4} &< 2^{(lb-t-3)-i}a_i + 2^{(lb-t-4)-(i-1)}a_{i-1} \\ &= 2^{(lb-t-3)-i}(a_i + a_{i-1}) < 2^{(lb-t-3)-i}n/(2^{lb-i-3}3) = n/(2^t 3). \end{aligned}$$

It follows from the same arguments as used for Region 2 that br$[a_i]$ will be pruned from the search tree unless n is a multiple of 5. □

It has been shown that if $n = 2^t m$, m odd, $\{b_i\}$ is bounding sequence (C), and $a_i < b_i$ for some i, then br$[a_i]$ can be pruned from the search tree and, furthermore, if n is not a multiple of 5, and $a_i + a_{i-1} < b_{i+1}$, then br$[a_i]$ can be pruned from the search tree provided that $n \ne 2^{lb-i}a_i$ for some step i in Region 1. If n is a multiple of 5 and if $\{b_i\}$ is bounding sequence (A), then if $a_i + a_{i-1} < b_{i+1} = n/2^{lb-(i+1)}$ for some i, br$[a_i]$ can be pruned from the search tree if there exists a step $s > i$ that is not a doubling since $a_s \le a_{s-1} + a_{s-2} \le 2^{(s-1)-i}a_i + 2^{(s-2)-(i-1)}a_{i-1} = 2^{(s-1)-i}(a_i + a_{i-1})$ from which it follows that $n < n$. Thus, when n is a multiple of 5, Theorem 4 holds for n provided that bounding sequence (C) is replaced by bounding sequence (A).

8. Pruning bounds applied. When the sequence $\{b_i\}$ is used as a class 1 bounding sequence, it will be called a *vertical* bounding sequence, and when it is used as a class 2 bounding sequence, it will be called a *slant* bounding sequence. If n is not a multiple of 5, then bounding sequence (C) can be used for both the vertical and slant bounds. The algorithm for generating addition chains checks to see: (1) if $a_i \ge b_i$; if this condition is satisfied, it next checks: (2) to see that if $n \ne 2^{lb-i}a_i$ for i in Region 1; then is $a_i + a_{i-1} \ge b_{i+1}$. If these conditions are met, then the partial chain $1 = a_0, a_1, \ldots, a_i$ is not rejected and candidates for a_{i+1} are considered. If (1) fails, then br$[a_i]$ is pruned from the search tree, and if (1) passes but (2) fails, br$[a_i]$ is pruned from the search tree.

If n is a multiple of 5, the same strategy is employed; however, bounding sequence (C) is used for the vertical bounds while bounding sequence (A) is used for the slant bounds.

TABLE 1

Value of n						
Test	23	127	191	568	607	1,015[2]
no bds	293	1,890,353 time: 20.9	24,383,588 time: 327.4			
V (A)	193	330,791 time: 1.6	2,805,121 time: 15.1	1,632,557 time: 8.57	63,548,562 time: 372.4	5,901,139 time: 32.0
V (C)	153	208,318 time: 1.0	1,789,703 time: 9.8	1,587,969 time: 8.18	31,649,325 time: 192.1	2,146,835 time: 12.1
V/S (A)	155	241,396 time: 1.0	2,025,178 time: 9.6	964,319 time: 4.39	40,847,534 time: 203.3	3,218,491 time: 14.6
V/S (C)	137	174,788 time: 0.82	1,482,328 time: 7.5	954,573 time: 4.28	24,823,053 time: 139.8	1,900,148 time: 9.9

The effect of these pruning bounds can be measured in terms of time required to search for chains as well as in terms of the number of "pops" off the stack. In the latter case, what is being popped off the stack for a given step i is the next candidate for a_i. The pruning bounds eliminate certain branches of the search tree, and the number of "pops" will diminish as the pruning bounds are put into effect. Several cases are considered for given values of n. The first case (no bds) has no pruning bounds. Case V (A) implements vertical pruning bounds using bounding sequence (A) while case V (C) implements vertical bounds using bounding sequence (C), etc. Case V/S (A) shows the effect of using bounding sequence (A) for both vertical and slant bounds. Case V/S (C) uses bounding sequence (C) for both vertical and slant bounds and when $n \equiv 0 \pmod 5$ replaces bounding sequence (C) with bounding sequence (A) for the slant bounds. Table 1 shows results for $n = 23$, 127, 191, 568, 607, and 1,015. These are interesting numbers from the standpoint of not giving trivial values yet not requiring so much time as to be unreasonable. The times, of course, are relative to the computer being used and are significant primarily in how they relate to each other as opposed to their actual values. It should be noted that when n is odd, bounding sequence (C) reduces to bounding sequence (B).

Table 1 records the number of "pops" that occur in an algorithm that finds all addition chains of minimal length for the given values of n. Also recorded is the time in seconds taken by each search. The times for $n = 23$ are too small to be significant.

A good check that the bounding sequences do not cut too much from the search tree is the observation that NMC(n) remains the same in all cases. As can be seen, there is significant improvement in going from V(A) to V/S(C).

9. Pruning bounds (class 1 further refined); the $2^i + 1$ phenomenon. For n odd, bounding sequence (B) can be improved as a vertical (class 1) bounding sequence depending on the multiplicative nature of n. If n is not a multiple of an integer of the form $2^i + 1$, then the following sequence can be used for the vertical bounding sequence.

$$\text{(D)} \qquad b_i = \begin{cases} \lceil n/(2^{lb-i-1}+1) \rceil & 0 \le i \le lb-1, \\ n & i = lb. \end{cases}$$

Suppose $a_i < \lceil n/(2^{lb-i-1}+1) \rceil$. Then $a_i < n/(2^{lb-i-1}+1)$. The cases $i = lb-1$ and $i = lb-2$ have been established previously. Let $i = lb-k$ for some k such that

[2]For $n = 1{,}015$, in the V/S (C) case bounding sequence (A) is used for the slant bounds since n is divisible by 5.

$3 \le k \le lb$.

$\boldsymbol{k = 3}$. $a_{lb-3} < n/5$. Since n is odd, step lb is not doubling. Yet it must be a star step. If $n = a_{lb} = a_{lb-1} + a_{lb-3}$, then $n \le 5a_{lb-3} < n$. If $n = a_{lb} = a_{lb-1} + a_{lb-2}$, then if $a_{lb-1} = 2a_{lb-2}$, it follows that 3 divides n, and if $a_{lb-1} = a_{lb-2} + a_{lb-3}$, then $n = a_{lb} = 2a_{lb-2} + a_{lb-3} \le 5a_{lb-3} < n$.

$\boldsymbol{k \ge 4}$. $a_{lb-k} < n/(2^{k-1}+1)$. If $n = a_{lb} = \boldsymbol{a_{lb-1} + a_{lb-k}}$, then

$$n = a_{lb} = a_{lb-1} + a_{lb-k} \le 2^{(lb-1)-(lb-k)} a_{lb-k} + a_{lb-k} = (2^{k-1}+1)a_{lb-k} < n.$$

Suppose $n = a_{lb} = \boldsymbol{a_{lb-1} + a_{lb-j}}$ for some $j, 2 \le j < k$. If $a_{lb-1} = 2^{j-1}a_{lb-j}$, then $2^{j-1}+1$ divides n. If $a_{lb-1} \ne 2^{j-1}a_{lb-j}$, then let s be the first integer greater than or equal to 1 such that step $lb - s$ is a nondoubling. Then $1 \le s < j$ and $n = a_{lb} = a_{lb-1} + a_{lb-j} = 2^{s-1}a_{lb-s} + a_{lb-j}$. This implies that

$$\begin{aligned} n &= 2^{s-1}a_{lb-s} + a_{lb-j} \le 2^{s-1}(a_{lb-s-1} + a_{lb-s-2}) + 2^{lb-j-(lb-k)}a_{lb-k} \\ &\le 2^{s-1}(2^{k-s-1}a_{lb-k} + 2^{k-s-2}a_{lb-k}) + 2^{k-j}a_{lb-k} \\ &= (2^{k-2} + 2^{k-3} + 2^{k-j})a_{lb-k}. \qquad \text{(Note: } s+2 \le k\text{)}. \end{aligned}$$

If $j \ge 3$, then $2^{k-2} + 2^{k-3} + 2^{k-j} < 2^{k-1} + 1$, and $n < n$. The only possibility is $j = 2$.

Consider $n = a_{lb} = \boldsymbol{a_{lb-1} + a_{lb-2}}$. Since $1 \le s < 2$, it follows that $s = 1$, and $a_{lb-1} \le a_{lb-2} + a_{lb-3}$. There are two cases for step $lb - 1$.

(i) $a_{lb-1} = a_{lb-2} + a_{lb-h}$. (Note: $h < k$. If $h = k$, then $n < n$.)

If $h = 2$, then 3 divides n. Thus, $a_{lb} = a_{lb-1} + a_{lb-2} = 2a_{lb-2} + a_{lb-h}$, where $h \ge 3$. Suppose there is a nondoubling between step $lb - h$ and step $lb - 2$. Suppose $lb - p$ is the first nondoubling going back in the chain from step $lb - 2$. Then $2 \le p < h$.

$$\begin{aligned} a_{lb} &= a_{lb-1} + a_{lb-2} = 2a_{lb-2} + a_{lb-h} = 2(2^{p-2}a_{lb-p}) + a_{lb-h} \\ &\le 2(2^{p-2}(a_{lb-p-1} + a_{lb-p-2})) + a_{lb-h}. \\ &\le 2^{p-1}(2^{k-p-1}a_{lb-k} + 2^{k-p-2}a_{lb-k}) + 2^{k-h}a_{lb-k} \\ &= (2^{k-2}a_{lb-k} + 2^{k-3}a_{lb-k}) + 2^{k-h}a_{lb-k} \\ &= (2^{k-2} + 2^{k-3} + 2^{k-h})a_{lb-k} < n \quad \text{since } h \ge 3. \end{aligned}$$

Thus, $a_{lb-2} = 2^{h-2}a_{lb-h}$ and $a_{lb} = a_{lb-1} + a_{lb-2} = 2a_{lb-2} + a_{lb-h} = 2(2^{h-2}a_{lb-h}) + a_{lb-h} = (2^{h-1}+1)a_{lb-h}$. This implies that $2^{h-1}+1$ divides n.

(ii) $a_{lb-1} = 2a_{lb-3}$. $n = a_{lb} = a_{lb-1} + a_{lb-2} = a_{lb-2} + 2a_{lb-3}$.

Then $n = a_{lb-2} + 2a_{lb-3} \le 2^{k-2}a_{lb-k} + 2(2^{k-3}a_{lb-k}) = 2^{k-1}a_{lb-k} < 2^{k-1}(n/(2^{k-1} + 1)) < n$.

This establishes the result that if n is not divisible by an integer of the form $2^i + 1, i \ge 0$, then the sequence (D) is a valid class 1 bounding sequence when generating addition chains for an integer n.

THEOREM 5. *If $2^j+1, j \ge 0$ does not divide n and $\{b_i\}$ denotes bounding sequence* (D), *then if $a_i < b_i$ for some i, $\mathrm{br}[a_i]$ can be pruned from the search tree.*

It appears that this result can generalized for $n = 2^t m$ (m odd) using the following bounding sequence.

$$\text{(E)} \qquad b_i = \begin{cases} \lceil n/2^t(2^{lb-t-(i+1)} + 1) \rceil & 0 \le i \le lb - t - 2, \\ \lceil n/2^{lb-i} \rceil & lb - t - 1 \le i \le lb. \end{cases}$$

Theorem 5 then would be restated substituting $j > 0$ for $j \ge 0$ and bounding sequence (E) for bounding sequence (D).

TABLE 2

Test	Value of n					
	23	127	191	568	607	1015[3]
no bds	293	1,890,353 time: 20.9	24,383,588 time: 327.4			
V (A)	193	330,791 time: 1.6	2,805,121 time: 15.1	1,632,557 time: 8.57	63,548,562 time: 372.4	5,901,139 time: 32.0
V (C)	153	208,318 time: 1.0	1,789,703 time: 9.8	1,587,696 time: 8.18	31,649,325 time: 192.1	2,146,835 time: 12.1
V (D)	153	186,506 time: 0.9	1,566,167 time: 8.95		25,381,280 time: 164.4	
VBEST	94	170,294 time: 0.9	1,440,524 time: 8.84	1,245,559 time: 8.29	23,316,309 time: 162.1	1,216,206 time: 8.24
V/S (A)	155	241,396 time: 1.0	2,025,178 time: 9.6	964,319 time: 4.39	40,847,534 time: 203.3	3,218,491 time: 14.6
V/S (C)	137	174,788 time: 0.82	1,482,328 time: 7.5	954,573 time: 4.28	24,823,053 time: 139.8	1,900,148 time: 9.9
V(D)/S (C)	137	166,120 time: 0.82	1,384,000 time: 7.36		22,486,525 time: 134.4	
VBEST/S (C)	92	150,522 time: 0.82	1,262,492 time: 7.09	792,751 time: 4.28	20,485,705 time: 130.4	1,087,144 time: 6.9

The numbers $23, 127, 191$, and 607 in Table 1 satisfy the hypothesis of Theorem 5. Table 2 shows results of using bounding sequence (D) for the vertical bounding sequence. Certain rows of the table use what is called VBEST for the vertical bounding sequence and bounding sequence (C) for slant bounds. VBEST is a bounding sequence that is obtained à posteriori by running the program and generating all minimal addition chains for n. For each i, b_i is the minimal a_i found among all the minimal addition chains for n. It is the best possible vertical bounding sequence, since lowering any b_i will cut out some of the minimal chains for n. In practice it is not a good bounding sequence since it cannot be determined until all the minimal addition chains have been found, but it is a good measure by which to judge the other vertical bounding sequences. As previously noted, bounding sequence (C) reduces to bounding sequence (B) when n is odd.

The V(D)/S (C) bounding sequence compares favorably with the VBEST/S (C) bounding sequence, and in all cases there is significant improvement over using just V(A) which is the obvious bounding sequence to try first.

While vertical bounding sequence (D) leads to a significant improvement in pruning the search tree, this sequence can only be used when n is not a multiple of an integer of the form $2^i + 1$. In other words, if n is a multiple of an integer of the form 2^i+1, then less aggressive pruning bounds must be used. More branches on the search tree must be explored. The search for minimal addition chains for such numbers is, in a sense, less efficient. Primes, on the other hand, (unless they are Fermat primes) can use bounding sequence (D) which results in a more efficient search.

A summary of the bounding sequences is included in Table 3.

10. Some test results. Algorithms for generating minimal length addition chains are significant from the standpoint of what their implementation in a computer program and the subsequent generation of data reveal about the addition chain

[3]For $n = 1015$, in the V/S (C) and VBEST/S(C) cases bounding sequence (A) is used for the slant bounds since n is divisible by 5. Also, sequence (D) cannot be used for vertical bounds since n is a multiple of 5.

TABLE 3

Bounding sequence	Sequence	Comments
A	$b_i = \lceil n/2^{lb-i} \rceil, i = 0, \ldots, lb$	vertical/slant bounds
B	$b_i = \begin{cases} \lceil n/(3 \cdot 2^{lb-(i+2)}) \rceil, & 0 \le i \le lb-2 \\ \lceil n/2^{lb-i} \rceil, & lb-1 \le i \le lb \end{cases}$	n odd, vertical bounds, slant bounds for $n \neq 5k$
C	$b_i = \begin{cases} \lceil n/(3 \cdot 2^{lb-(i+2)}) \rceil, & 0 \le i \le lb-t-2 \\ \lceil n/2^{lb-i} \rceil, & lb-t-1 \le i \le lb \end{cases}$	$n = 2^t m$, m odd, vertical bounds, slant bounds if $n \neq 5k$ and $n \neq 2^{lb-i} a_i$ for $i \ge lb-t-2$
D	$b_i = \begin{cases} \lceil n/(2^{lb-i-1}+1) \rceil, & 0 \le i \le lb-1 \\ n, & i = lb \end{cases}$	n odd, vertical bounds $n \neq (2^j+1)k$, $j \ge 0$
E	$b_i = \begin{cases} \lceil n/2^t(2^{lb-t-(i+1)}+1) \rceil, & 0 \le i \le lb-t-2 \\ \lceil n/2^{lb-i} \rceil, & lb-t-1 \le i \le lb \end{cases}$	$n = 2^t m$, m odd, vertical bounds $n \neq (2^j+1)k$, $j > 0$

problem. Knuth [12] found that $l(191) = l(382)$. He found other integers as well for which $l(2n) = l(n)$. This was a somewhat surprising discovery. Utz [20] had asked if $l(n) < l(2n)$ for all $n > 0$, and it had supposedly been proved earlier by Jonquières [11] that $l(2n) = l(n) + 1$. Subsequently, it has been proved [17, 18] that there are infinite classes of integers for which $l(2n) = l(n)$. If $h(x)$ denotes the number of integers $n \le x$ for which $l(2n) = l(n)$, then computer generated data suggest the possibility that $h(x)$ is bounded below by some constant times x.

While Hansen [10] proved that there exist integers that do not admit star chains among their minimal addition chains, Knuth's computer calculations revealed $n =$ 12,509 as the first number that does not admit a star chain among its minimal chains. It is very likely that $\{2^m(81) + 17, m \ge 8\}$ is an infinite class of such integers.

The addition chain problem is exponential in nature [8]. Improving pruning algorithms enables one to generate the next level of data which can reveal properties that had not previously been observed. For example, $n = 49{,}593$ and $n = 49{,}594$ are adjacent integers, neither of which has a star chain among its minimal chains. $n = 13{,}818$ and $n = 27{,}578$ are even numbers for which $l(2n) = l(n)$. If $n = 2k$, this is an example of an integer for which $l(4k) = l(2k)$. Are there integers for which $l(8k) = l(4k)$? The pairs $n = 22{,}453$, $n = 22{,}455$ and $n = 25{,}019$, $n = 25{,}021$ have the property that $l(2n) = l(n)$. Are there consecutive integers with this property? $n = 29{,}479$ is an integer, all of whose minimal chains start with $1, 2, 3$ which verifies a conjecture of Chin and Tsai [5]. Such integers appear very rarely. The first integer with over 1,000,000 minimal addition chains is 15,126. In fact, NMC(15, 126) $= 1{,}047{,}580$.

Knuth found $c(r)$, the first integer for which $l(n) = r$, for $1 \le r \le 18$. The algorithms discussed in the paper were instrumental in determining that $c(19) =$ $18{,}287, c(20) = 34{,}303, c(21) = 65{,}131$. Flammenkamp and Bleichenbacher have extended this sequence to $c(22) = 110{,}591$, $c(23) = 196{,}591$, $c(24) = 357{,}887$, $c(25) =$ $685,951$, $c(26) = 1,176,431$, and $c(27) = 2,211,837$. See [12] and the online version of [15]. $n = 65{,}131$ is the first integer that requires six small steps in a minimal addition chain. The increase in the required number of small steps in an addition chain greatly increases the computing time. Table 4 shows how the computing time increases with the number of small steps while $\lambda(n)$ is held constant. It also shows how the computing time increases with $\lambda(n)$ while holding the number of small steps constant.

TABLE 4

n	binary n	$\lambda(n)$	number of small steps in chain	pops time
10,241	10100000000001	13	2	4,300 0.00 secs.
10,247	10100000000111	13	3	1,212,209 7.14 secs.
10,271	10100000011111	13	4	220,689,095 1441.00 secs.
20,487	101000000000111	14	3	1,682,449 10.33 secs.
20,494	101000000001110	14	3	2,886,936 15.60 secs.
40,967	1010000000000111	15	3	2,279,240 14.50 secs.
40,988	1010000000011100	15	3	5,286,601 28.23 secs.
81,927	10100000000000111	16	3	3,024,682 19.99 secs.
81,976	10100000000111000	16	3	8,593,425 46.20 secs.

Holding $\lambda(n)$ constant for odd n, while increasing the number of small steps, greatly increases the computing time. As the number of ones in the binary representation of n increases, the number of small steps increases in minimal addition chains for n. In fact $N(n)$, the number of small steps, appears to be bounded below by $\lceil \log_2 \nu(n) \rceil$. As can be seen, computation time is influenced much more strongly by the size of $\nu(n)$ than by the size of n. The minimal addition chains for $n = 1{,}048{,}577$ can be found much more quickly than the minimal addition chains for $n = 191$.

Increasing $\lambda(n)$, while holding the number of small steps constant, increases the computation time relatively slowly. Numbers of the form $n = 2^t m$ for m odd and $t \geq 1$ have longer computation times than odd integers with the same values of $\lambda(n)$ and $N(n)$. As t increases with $\lambda(n)$ and $N(n)$ held constant, the computing time increases. An examination of pruning bounds (C) shows that the bounds are less severe for integers with a large value of t.

The pruning bounds operate the best on odd integers not divisible by 5 and, in particular, on odd integers not divisible by integers of the form $2^i + 1, i \geq 1$.

Data generated by this algorithm have suggested theorems concerning NMC(n), some of which have been established in [19]. These mathematical proofs indirectly confirm the validity of the programs that generated the data that suggested the theorems.

11. Conclusion. The pruning bounds that have been developed significantly improve the efficiency of the search for minimal length addition chains for an integer. It is of interest to note that, for multiples of $2^i + 1$, the pruning bounds cannot be as tight. As can be seen, the best general vertical bounds compare favorably with the VBEST vertical bounds for the examples cited in Table 2.

Introducing slant bounds (bounds for $a_i + a_{i-1}$) increases the efficiency of the search. This class of bounds can be extended to bounds for $a_i + a_{i-2}$ and $a_{i-1} + a_{i-1}$ and perhaps other sums. It appears likely, however, that improvements in efficiency are slight and are negated by the time needed to run all the checks on the additional bounds.

The most famous unsolved problem in addition chains is the Scholz–Brauer con-

jecture which states that $l(2^n-1) \leq n+l(n)-1$. Brauer [4] proved that the conjecture is true when n includes a star chain among its minimal chains. Hansen [10] developed the concept of l^0-chains of which star chains are a proper subset. He proved that the Scholz–Brauer conjecture is true if n includes an l^0-chain among its minimal chains. It remains an open question as to whether every integer includes an l^0-chain among its minimal chains. Computer searches may prove useful in helping to settle this question.

Acknowledgments. This paper has benefited from fruitful discussions with Khalaf Haddad, a former student, and Dan Eilers, president of Irvine Compiler Corporation.

REFERENCES

[1] F. Bergeron, J. Berstel, S. Brlek, and C. Duboc, *Addition chains using continued fractions*, J. Algorithms, 10 (1989), pp. 403–412.
[2] F. Bergeron, J. Berstel, and S. Brlek, *Efficient computation of addition chains*, J. de Théor. Nombres Bordeaux, 6 (1994), pp. 21–38.
[3] J. Bos and M. Coster, *Addition chain heuristics*, in Proceedings CRYPTO '89, 1990, pp. 400–407.
[4] A. T. Brauer, *On addition chains*, Bull. Amer Math. Soc., 45 (1939), pp. 736–739.
[5] Y. H. Chin and Y.-H. Tsai, *Algorithms for finding the shortest addition chain*, in Proceedings National Computer Symposium, Kaoshiung, Taiwan, Dec. 1985, pp. 1398–1414.
[6] H. Dellac, *Question* 49, l'Interm. des Math., 1 (1894), p. 20.
[7] D. Dobkin and R. J. Lipton, *Addition chain methods for the evaluation of specific polynomials*, SIAM J. Comput., 9 (1980), pp. 121–125.
[8] P. Downey, B. Leong, and R. Sethi, *Computing sequences with addition chains*, SIAM J. Comput., 10 (1981), pp. 638–646.
[9] P. Erdős, *Remarks on number theory III on addition chains*, Acta Arith., 6 (1960), pp. 77–81.
[10] W. Hansen, *Zum Scholz-Brauerchen Problem*, J. Reine Angew. Math., 202 (1959), pp. 129–136 (in German).
[11] E. de Jonquières, *Question* 393, (A. Goulard), l'Interm. des Math., 2 (1985), pp. 125–126.
[12] D. E. Knuth, *The Art of Computer Programming*, vol. 2, 3rd ed., Addison–Wesley, Reading, MA, 1997, pp. 461–485.
[13] A. Scholz, *Aufgabe* 253, Jahresber. Deutsch. Math.-Verein., 47 (1937), pp. 41–42.
[14] A. Schönhage, *A lower bound for the length of addition chains*, Theoret. Comput. Sci., 1 (1975), pp. 1–12.
[15] N. J. A. Sloane and S. Plouffe, *The Encyclopedia of Integer Sequences*, Academic Press, San Diego, 1995.
[16] M. V. Subbarao, *Addition chains—some results and problems*, in Number Theory and Applications, R. A. Mollin, ed., Kluwer Academic Publishers, Dordrecht, 1989, pp. 555–574.
[17] E. G. Thurber, *The Scholz–Brauer problem on addition chains*, Pacific J. Math., 49 (1973), pp. 229–242.
[18] E. G. Thurber, *Addition chains and solutions of* $l(2n) = l(n)$ *and* $l(2^n - 1) = n + l(n) - 1$, Discrete Math., 16 (1976), pp. 279–289.
[19] E. G. Thurber, *Addition chains—an erratic sequence*, Discrete Math., 122 (1993), pp. 287–305.
[20] W. R. Utz, *A note on the Scholz–Brauer problem on addition chains*, Proc. Amer. Math. Soc., 4 (1953), pp. 462–463.